

Building A Simple Test Bed For WSN

Arathi S M

*Department of Computer Science and Engineering
BNMIT, Bengaluru, Karnataka, India*

Chaitra K V

*Department of Computer Science and Engineering
BNMIT, Bengaluru, Karnataka, India*

Deepthi Devaraj

*Department of Computer Science and Engineering
BNMIT, Bengaluru, Karnataka, India*

Jyothsna S Sastry

*Department of Computer Science and Engineering
BNMIT, Bengaluru, Karnataka, India*

Shashikala

*Department of Computer Science and Engineering
BNMIT, Bengaluru, Karnataka, India*

Abstract- Wireless communication is a rapidly growing technology dominating the language of communication. It plays an important role in every sector of the communications frontier and hence proves to be a powerful technology that allows exchange of information from anywhere in the world with the help of mobile devices. Local availability, coverage range, energy consumption, maintenance cost and performance are some of the concrete parameters that define the communication technologies. This paper provides the basic knowledge to build a simple test bed for a Wireless Sensor Network. The setup and configuration steps for the topology are suggested.

Keywords – Wireless, Communication, XBee, Arduino, Test Bed

I. INTRODUCTION

Wired networks in homes and organizations are being replaced by wireless networks owing to its wide range of applications. Wireless sensor network (WSN) is a wireless network with little or no infrastructure, consisting of a number of sensor nodes distributed in a deployment zone to monitor and obtain data about the environment. Examples of wireless networks include cell phone networks, Wi-Fi local networks and terrestrial microwave networks. WSN's can be classified as either unstructured or structured[1]. Unstructured WSN consists of a dense collection of sensor nodes that are deployed in an ad hoc manner. The deployed network is left unattended which makes it difficult to manage connections and detect failures. In a structured WSN, the sensor nodes are deployed in a pre-defined manner. Deployment of limited nodes in a structured WSN reduces the management cost and maintenance overhead.

The challenges concerned with the design of a wireless system must be addressed to enable the wireless applications of the future. Resource constraints of a WSN could be short communication range, limited processing, low bandwidth, limited energy, and storage in each node while the design constraints are application dependent and are based on the monitored environment. The environment plays a key role in determining the size of the network, the deployment scheme and the network topology.

WSN's have numerous advantages in supporting information exchange between people or devices. It is quite simple to relocate a communicating device as there is no additional cost of rewiring associated with such a move. The system need not be disrupted while adding or removing a communication device to or from the network. Other than

the initial outlay on setting up a wireless network, the cost of running and maintaining it is minimal. These factors of wireless technology make it suitable for the home and office environment.

II. WIRELESS TECHNOLOGY

The wireless technology is constantly emerging, sidelining the old technologies. A few important technologies[2] are Cellular Telephone systems, RF modem, Wireless LAN's, Broadband wireless access, Bluetooth and ZigBee. Cellular Telephone systems marked the beginning of wireless revolution. They provide a two way voice and data communication channel over any region. An RF module is a small electronic circuit that transmits and receives radio signals on one amongst a number of carrier frequencies. Electronic and designing of radio circuitry are based on RF modules. On the other hand, the wireless LAN's provide data over a small region. Worldwide Interoperability for Microwave Access (WiMAX) is an emerging broadband wireless technology that transfers data at a faster rate. This technology was once used by several mobile carriers to deliver wireless data to its customers. The low-cost low-power radio's are Bluetooth and ZigBee. Bluetooth provides a short range communication between wireless devices along with rudimentary network abilities. ZigBee is a protocol that uses the 802.15.4 standard[4] which is a simple, low cost and low power wireless communication technique used in embedded applications. The ZigBee radio specification is the most widely accepted technology when compared to Bluetooth. XBee is the name given to the radio communication module. The modules can be made to support ZigBee protocol by loading additional firmware.

These technologies can be implemented on a number of test beds, emulators and simulators. A test bed for WSN consists of sensor nodes deployed in a controlled environment. These test beds are designed to support the experimental research in real world setting. It provides a way to test protocol, algorithms, network issues and applications. Building a test bed for a particular WSN application is quite expensive, time consuming and difficult. A computer system can be made to imitate the behavior of another system with the help of an emulator. Hardware or software emulators allow the host system to run the peripheral software devised for the guest system. Emulators allow a system to work with a software that is exclusive to another system. Real nodes being emulated meets the performance standard.

Protocols, schemes and algorithms can be evaluated at a large scale at the beginning of the design phase by using simulators[5]. The cost associated with simulating thousands of nodes is very low and can be done within a stipulated time frame. Simulation should not be used when the given problem can be solved analytically or when direct experiments can be performed.

III. METHODOLOGY

Wireless Communication involves data transfer between two or more remote nodes without the use of physical conductors. This paper describes the building of a simple test bed in stages. The data is wirelessly transmitted between two nodes that are powered using the laptops or 9V battery. Data sensed by the sensor node is processed by the microcontroller and is transmitted to the remote coordinator wirelessly.

Components Required :

Arduino Uno: Arduino Uno is a microcontroller board based on the ATmega328. It can be powered using a battery (9V) or by a computer using a USB cable. It contains a USB connection, a reset button, a 16MHz ceramic resonator, 14 digital input/output pins in addition to 6 analog inputs. "Uno" means one in Italian and is named to mark the release of Arduino 1.0. Several revisions to the Uno board incorporate new features. For instance, the revision 2 to the Uno board facilitates DFU mode of operation while the revision 3 allows the shields to adapt to the voltage provided by the board and includes a stronger RESET circuit. Arduino Uno serves as the reference model for the Arduino platform. The specifications of Arduino Uno[3] is as shown in table 1:

Name	Parameter values
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA

DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by boot
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g
XBee ZB Radio frequency	2.4 GHz

Table 1 Specifications of Arduino Uno

XBee Explorer USB: The initial step to communicating with the XBee module is to pick a suitable interface which facilitates communication. The XBee Explorer functions as the gateway between the computer and the XBee module. A number of Explorers are available for this purpose. The XBee Explorer USB is the most popularly used interface among them.

This unit is designed to work with Series 1 and Series 2, standard and Pro version of the XBee modules. It is equipped with a mini-B USB connector. The XBee module should be plugged into the XBee Explorer and a mini USB cable should be attached to the Explorer to establish direct access to the serial and programming pins of the XBee module. This board contains a USB-to-Serial converter which helps in translating the data between the computer and the XBee. In addition, it is equipped with a RESET button and voltage regulator to provide the XBee with adequate power. The four LEDs (RX, TX, RSSI and Power indicator) help in debugging the XBee. It is necessary to install the Explorer drivers. Successfully installed drivers have a unique port number assigned to them (COM#) regardless of the Operating system in the computer.

XBee Module: XBees are the most popularly used wireless transceivers because of the two key features they possess.

1. Flexibility: They are compatible with computers and microcontrollers like Arduino because they can send and receive data over a serial port.
2. Configurability: A pair of Xbee devices can be used for exchanging data or a large number of these devices can be configured to transfer data across a mesh network.

They are available in a variety of ranges, series and frequencies. They can be used in smart home applications that monitor and remotely control the lighting and the temperature in rooms.

Temperature Sensor LM35/LM335: It is an integrated circuit sensor that can be used to measure temperature with an electrical output proportional to the temperature. An important characteristic of this sensor is that it poses a low self heating capability causing less than 0.1 °C temperature rise in still air.

The transition from wired to wireless communication is facilitated by the XBee module. Using XBee devices for communication is advised for low-cost applications that require reliable, bi-directional communication at moderate speeds.

IV. INITIAL SETUP

X-CTU being a multipurpose application can be used with Linux, Windows and MacOS. It is a free software provided by Digi to connect, configure and test the remote XBee.

To begin with, install X-CTU on the wine platform in Ubuntu. This application is required to change the configuration settings of XBee series 2. To install wine directory, the command *winefile* is executed. A computer communicates with other devices using serial data port or communication port (COM). The USB ports have to be linked to the COM ports. This is done by creating symbolic links using the command *ln -s*. Figure 1 depicts the same. Here the /dev/ttyUSB2 is the USB port which is linked to the COM port 3.



Fig. 1 Creation of symbolic links

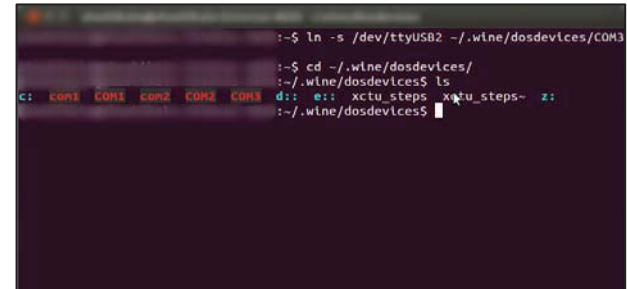


Fig. 2 Change to dosdevices directory

Change the directory to *dosdevices* using the command *cd* as shown in figure 2. Registry Editor in Ubuntu is used to perform advanced troubleshooting and environment specific modifications. In order to perform the required modifications on the registry keys, open the wine Registry Editor using the command *regedit* as shown in figure 3.

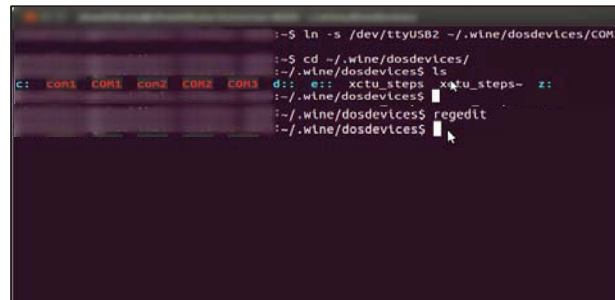


Fig. 3 Open Registry Editor

Create the file *SERIAL* under *Enum* folder present inside *CurrentControlSet* which can be viewed on expanding the *System* folder as seen in figure 4. Create a key for every COM port in the *SERIAL* folder. Each COM port is assigned a ClassGUID with the value {4D36E978-E325-11CE-BFC1-08002BE10318} and a FriendlyName. To connect more than one device, every COM port has to be configured. Use of single port ID can connect only one device, while multiple port ID can connect two or more devices. For Multiple Ports, the data of ClassGUID is {50906cb8-ba12-11d1-bf5d-0000f805f530} as shown in figure 5.

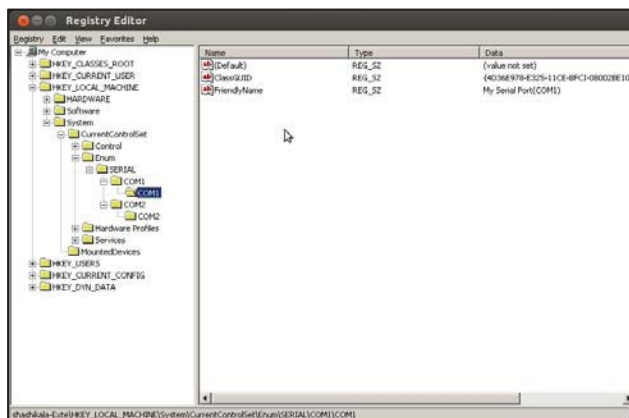


Fig. 4 Registry Editor for first COM port

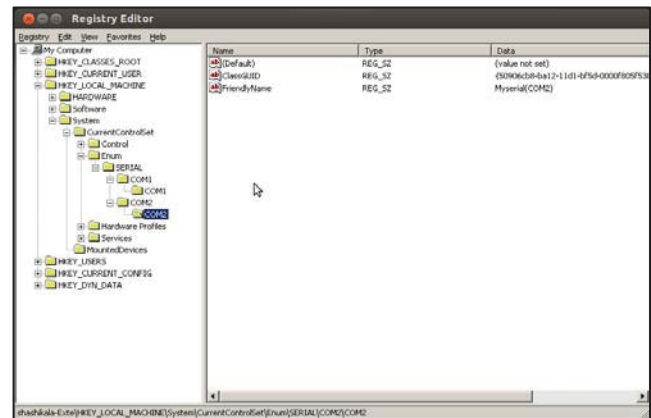


Fig. 5 Registry Editor for second COM port

Plug in the USB-Serial Port adaptor to one of the USB ports. Run *dmesg* as shown in figure 6.

The following lines must appear in the output :

usb X-1: new full speed USB device using uhci_and address 2

usb X-1: configuration #1 chosen from 1 choice

```
[ 187.452031] sd 4:0:0:0: [sdb] Assuming drive cache: write through
[ 187.452045] sd 4:0:0:0: [sdb] Attached SCSI removable disk
[ 327.109659] type=1400 audit(143470709.576:29): apparmor="DENIED" operation="capable" parent=1 profile="/usr/sbin/cupsd" pid=634 comm="cupsd" pid=634 comm="cupsd" capability=36 capname="block_suspend"
[ 1143.920182] usb 5-1: new full-speed USB device number 2 using uhci_hcd
[ 1144.082292] usb 5-1: New USB device found, idVendor=10c4, idProduct=ea60
[ 1144.082301] usb 5-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1144.082308] usb 5-1: Product: CP2102 USB to UART Bridge Controller
[ 1144.082313] usb 5-1: Manufacturer: Silicon Labs
[ 1144.082319] usb 5-1: SerialNumber: 0001
[ 1144.153367] usbcore: registered new interface driver usbserial
[ 1144.153390] usbcore: registered new interface driver usbserial_generic
[ 1144.153407] USB Serial support registered for generic
[ 1144.153416] usbserial: USB Serial Driver core
[ 1144.166665] usbcore: registered new interface driver cp210x
[ 1144.166688] USB Serial support registered for cp210x
[ 1144.166728] cp210x 5-1:1.0: cp210x converter detected
[ 1144.276179] usb 5-1: reset full-speed USB device number 2 using uhci_hcd
[ 1144.424499] usb 5-1: cp210x converter now attached to ttyUSB0
[ 1170.457372] type=1400 audit(1434701552.924:30): apparmor="DENIED" operation="capable" parent=1 profile="/usr/sbin/cupsd" pid=634 comm="cupsd" pid=634 comm="cupsd" capability=36 capname="block_suspend"
~/wine/dosdevices$
```

Fig. 6 dmesg output

```
[ 1144.153367] usbcore: registered new interface driver usbserial
[ 1144.153390] usbcore: registered new interface driver usbserial_generic
[ 1144.153407] USB Serial support registered for generic
[ 1144.153416] usbserial: USB Serial Driver core
[ 1144.166665] usbcore: registered new interface driver cp210x
[ 1144.166688] USB Serial support registered for cp210x
[ 1144.166728] cp210x 5-1:1.0: cp210x converter detected
[ 1144.276179] usb 5-1: reset full-speed USB device number 2 using uhci_hcd
[ 1144.424499] usb 5-1: cp210x converter now attached to ttyUSB0
[ 1170.457372] type=1400 audit(1434701552.924:30): apparmor="DENIED" operation="capable" parent=1 profile="/usr/sbin/cupsd" pid=634 comm="cupsd" pid=634 comm="cupsd" capability=36 capname="block_suspend"
~/wine/dosdevices$ lsusb
Bus 001 Device 002: ID 5986:0105 Acer, Inc
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 007 Device 002: ID 0a5c:2101 Broadcom Corp. Bluetooth Controller
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
~/wine/dosdevices$
```

Fig. 7 Execution of lsusb

Execute the *lsusb* command after unplugging the device. Ensure that the lines similar to the following lines appear in the output :

Bus 003 Device 001: ID 0000:0000

Bus 002 Device 007: ID 03f0:4f11 Hewlett-Packard

Bus 002 Device 006: ID 05e3:1205 Genesys Logic, Inc. Afilius Optical Mouse H3003

Bus 002 Device 004: ID 15d9:0a33

Run the command *lsusb* again once the USB-Serial adaptor is plugged in. An additional line with Vendor ID and Product ID will appear in the output as highlighted in the figure 8.

To load the module *usbserial* into the linux kernel, the command *sudo modprobe* with arguments Vendor ID and Product ID is executed as shown in figure 9.

```
Bus 001 Device 002: ID 5986:0105 Acer, Inc
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 007 Device 002: ID 0a5c:2101 Broadcom Corp. Bluetooth Controller
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
~/wine/dosdevices$ lsusb
Bus 001 Device 002: ID 5986:0105 Acer, Inc
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 005 Device 003: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x Composite Device
Bus 007 Device 002: ID 0a5c:2101 Broadcom Corp. Bluetooth Controller
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
~/wine/dosdevices$
```

Fig. 8 Execution of lsusb to find the Vendor ID and Product ID

```
cupsd" capability=36 capname="block_suspend"
~/wine/dosdevices$ lsusb
Bus 001 Device 002: ID 5986:0105 Acer, Inc
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 007 Device 002: ID 0a5c:2101 Broadcom Corp. Bluetooth Controller
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
~/wine/dosdevices$ lsusb
Bus 001 Device 002: ID 5986:0105 Acer, Inc
Bus 002 Device 003: ID 0781:556b SanDisk Corp.
Bus 005 Device 003: ID 10c4:ea60 Cygnal Integrated Products, Inc. CP210x Composite Device
Bus 007 Device 002: ID 0a5c:2101 Broadcom Corp. Bluetooth Controller
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 006 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 007 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
~/wine/dosdevices$ sudo modprobe usbserial vendor=10c4 product=ea60
~/wine/dosdevices$
```

Fig. 9 Execution of sudo modprobe

To ensure that the *usbcore* has registered a new interface, run the command *dmesg* again as shown in figure 10.

```

1683.568247] cfg80211: Disabling freq 5560 MHz
1683.568249] cfg80211: Disabling freq 5580 MHz
1683.568250] cfg80211: Disabling freq 5600 MHz
1683.568252] cfg80211: Disabling freq 5620 MHz
1683.568254] cfg80211: Disabling freq 5640 MHz
1683.568256] cfg80211: Disabling freq 5660 MHz
1683.568257] cfg80211: Disabling freq 5680 MHz
1683.568259] cfg80211: Disabling freq 5700 MHz
1683.568263] cfg80211: Regulatory domain changed to country: IN
1683.568264] cfg80211: (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
1683.568267] cfg80211: (2402000 KHz - 2482000 KHz @ 40000 KHz), (N/A, 2000 mBm)
1683.568269] cfg80211: (5170000 KHz - 5250000 KHz @ 40000 KHz), (N/A, 2000 mBm)
1683.568272] cfg80211: (5250000 KHz - 5330000 KHz @ 40000 KHz), (N/A, 2000 mBm)
1683.568274] cfg80211: (5735000 KHz - 5835000 KHz @ 40000 KHz), (N/A, 2000 mBm)
0.654028] usbcore: registered new interface driver usbfs
0.654028] usbcore: registered new interface driver hub
0.654028] usbcore: registered new device driver usb
1.005940] usbcore: registered new interface driver libusual
18.315089] usbcore: registered new interface driver uvcvideo
19.576371] usbcore: registered new interface driver btusb
186.433770] usbcore: registered new interface driver usb-storage
186.439591] usbcore: registered new interface driver uas
1144.153367] usbcore: registered new interface driver usbserial
1144.153390] usbcore: registered new interface driver usbserial_generic
1144.166605] usbcore: registered new interface driver cp210x
~/.wine/dosdevices$ dmesg | grep usbcore
~/.wine/dosdevices$

```

Fig.10 Execution of *dmesg / grep usbcore*

In X-CTU, new user-defined ports are added by clicking on the User Com Ports tab and the name of the port is typed in the field provided in the Com Port Number. The user interface of X-CTU is as shown in figure 11.

On selecting the recently added COM port, click on the Test/Query button. This communicates with XBee module connected to that port as shown in figure 12.

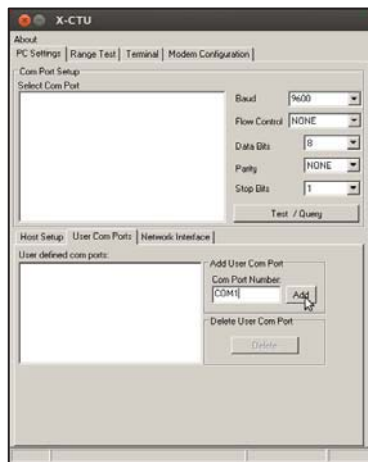


Fig.11 X-CTU interface

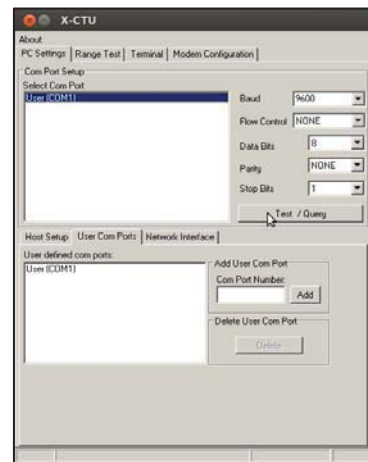


Fig.12 Test/Query the XBee module

The modem type and modem firmware version is displayed on communicating with the XBee module as shown in figure 13.

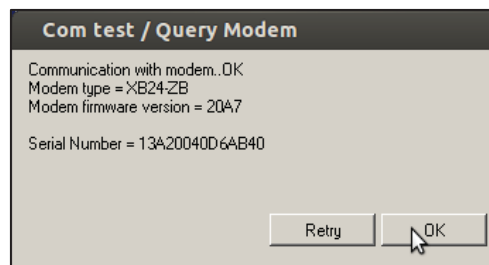


Fig. 13 COM Test/Query Modem

On successful communication with the XBee module, the parameters of the module to be read can be seen in the modem configuration tab. The “Always Update Firmware” has to be checked and the parameters can be read by clicking on the read button as shown in figure 14.

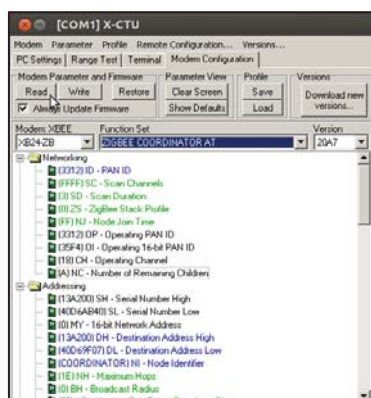


Fig. 14 Setting XBee module parameters

The parameters can be updated/set in this window by setting the modem type and Function set. The value for every parameter can also be changed depending on the user requirement while ensuring the communicating XBee modules have the same operating channel as seen in figure 14 which can be displayed by reading the individual parameter. The Personal Area Network(PAN) ID can be set as any value between 0 and 0xFFFF, while Node Identifier is a user defined name that explicitly mentions the job of the XBee. Communication between XBee modules is possible only if they have the same PAN ID. These set parameters are written on to the XBee by clicking the write button. The parameter of the Xbee has to be set as shown in table 2 :

SETTING	ACRONYM	XBEE1	XBEE2
Channel	CH	C	C
PAN ID	ID	PID	PID
Destination Address High	DH	SH2	SH1
Destination Address Low	DL	SL2	SL2

SH : Serial Number High SL : Serial Number Low

Table 2 Parameters of XBee

Communication can be seen on clicking the Terminal tab.

The wireless communications between two configured XBee's operating in the same channel are powered. The signals sensed by a sensor is processed and sent to the other sensor wirelessly. The communication between the two sensors can be seen on the laptop screens respective to the XBee's connected as shown in figure 15.1 and 15.2.

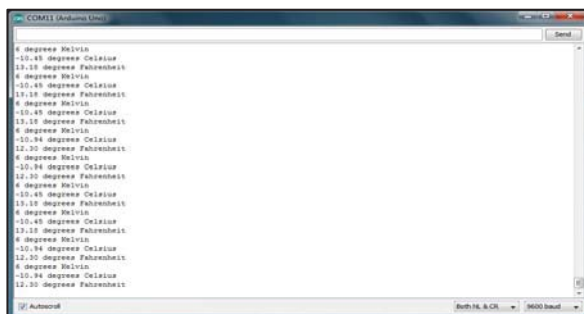
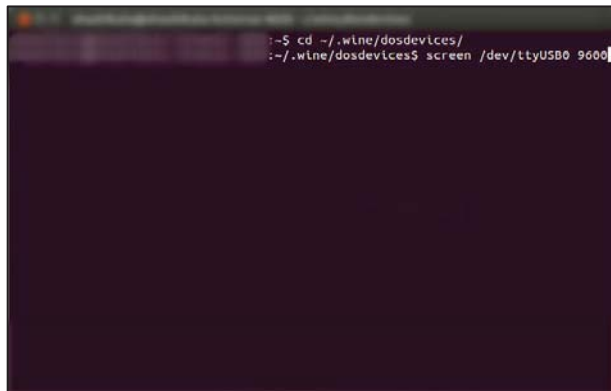


Fig. 15.1 Data sent by XBee



Fig. 15.2 Data received by XBee 2

If the X-CTU terminal does not display the received data from the sensor, the data can be viewed on the terminal of the receiving XBee by executing the command *screen* with USB port and baud rate as the parameters as shown in figure 16.

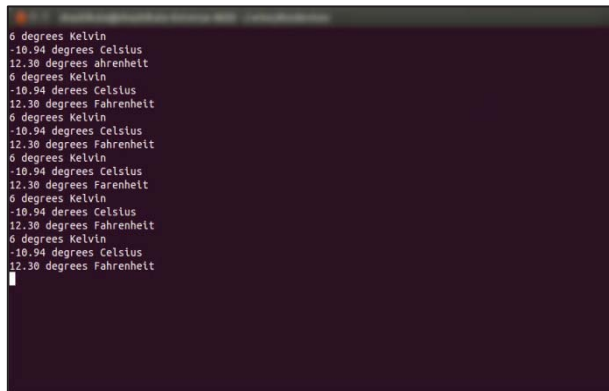


```

~$ cd ~/.wine/dosdevices/
~/.wine/dosdevices$ screen /dev/ttyUSB0 9600

```

Fig. 16 screen command



```

6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit
6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit
6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit
6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit
6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit
6 degrees Kelvin
-10.94 degrees Celsius
12.30 degrees Fahrenheit

```

Fig.17 Received data

The data will be displayed on the terminal as shown in figure 17.

IV.CONCLUSION

The test bed for the network built by the afore mentioned steps facilitates effective communication over large distances. This arrangement can be deployed in applications where the communication devices are not wired. Care should be taken so that the communicating devices are not placed outside the coverage range. The devices placed beyond the coverage range cannot be communicated thus disrupting the communication link. The suggested topology can be enhanced by increasing the number of XBee modules to support applications that demand communications across longer distances.

REFERENCES

- [1] Jennifer Yick, Biswanath Mukherjee, Dipak Ghosal, "Wireless sensor network survey". Computer Networks 52 (2008) 2292–2330
- [2] V.Abinayaa, Anagha Jayan, "Case Study on Comparison of Wireless Technologies in Industrial Applications". International Journal of Scientific and Research Publications, Volume 4, February 2014
- [3] www.arduino.cc
- [4] Demystifying 802.15.4 and ZigBee (White Paper)
- [5] Fei Yu, "A Survey of wireless sensor network simulation tools". 2003.